

## 附录 A Stata 入门

Stata 是目前较为流行的统计分析软件之一，具有强大的数据处理、统计和计量分析功能。本附录简要介绍 Stata 的基础知识以及利用 Stata 进行数据处理和绘图的基本操作，为读者学习 Stata 提供参考。本附录的写作主要参考许琪 (2021) 和毛新述 (2022) 以及公众号“连享会”的相关推送，感兴趣的读者可进一步阅读相关资料。

在本附录中，“.”表示 Stata 环境中的命令提示符，也用来表示缺失值，也参与构成因子变量等特殊函数，结合上下文容易判断“.”的具体含义。

### A.1 初识 Stata

在使用 Stata 进行数据分析之前，通常需要设置当前工作路径，在 Stata 中，我们可以通过 `cd "pathname"` 命令设置工作路径，用户可根据需要设置自己的 *pathname*。

#### 工作界面

对于已经完成安装和激活的 Stata 软件，打开以后会弹出如图 A.1 所示的工作界面。在工作界面中央所占面积最大的是结果窗口，主要显示用户的输入命令和输出结果。在结果窗口正下方的是命令窗口，在该窗口输入命令后按回车键即可执行命令。在工作界面左边的窗口是历史窗口，主要显示已经执行过的命令，其中黑色代表被正确执行，红色代表出错的命令。位于工作界面右上角的窗口是变量窗口，能够显示已经读入的数据中所有变量的变量名称和变量标签。位于工作界面右下角的窗口是属性窗口，主要显示已读入的数据和变量的属性。

此外，Stata 工作界面的上方有一排选项，即“文件”、“编辑”、“数据”、“图形”等，我们称之为菜单栏。在菜单栏的下方有一系列工具按钮，我们称之为工具栏，工具栏提供了 Stata 常用功能的快捷操作。

#### 命令帮助与安装

在学习新命令时，可以通过 `help` 命令查阅帮助文档快速获取命令的使用方法，例如，想要查询命令 `xtreg` 的使用方法，在命令窗口输入 `help xtreg` 即可。

作为一个开放性软件，用户可以对 Stata 官方提供的命令进行修改并编写一些新命令，这些命令就是外部命令，当运行某命令时发现 `command ... is unrecognized` 时，其中，... 为该命令名称，则意味着 Stata 并没有安装这一命令，此时需要利用 `ssc install ...` 进行安装。用户可通过输入 `ssc hot` 查看当前最火热的外部命令，同时可使用 `ssc uninstall` 卸载已经安装好的命令。

#### Stata 的命令格式

Stata 的命令格式通常为如下一般句法规则 (syntax) 的子集：

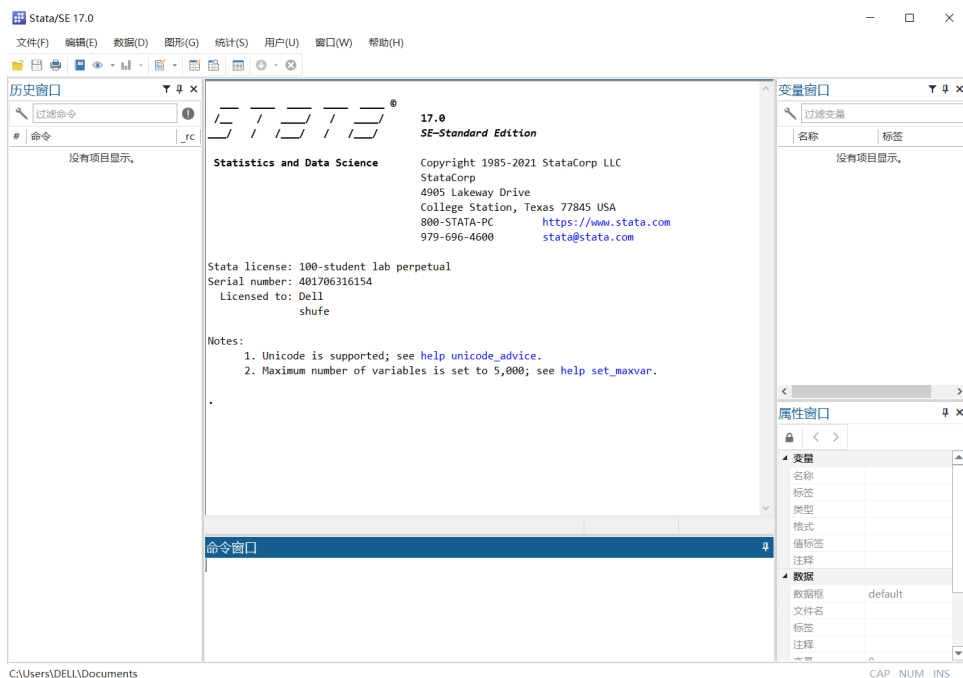


图 A.1: Stata 的工作界面

```
1 . [prefix:] command [varlist] [= exp] [if] [in] [weight] [using
    filename] [, options]
```

其中,

- *prefix*: *command* 的前缀, 用来重复执行 *command*, 改变 *command* 的输入或输出;
- *command*: Stata 的某个命令;
- *varlist*: 变量名列表;
- *exp*: 数学表达式;
- *weight*: 权重表达式;
- *filename*: *command* 使用的 (数据) 文件;
- *options*: 一个或多个适用于 *command* 的选项。

此外, 本附录中提到 Stata 的命令格式时约定: 中括号表示的是可选项; 若使用打印机字体 (如: *using*) 显示的部分, 则应依照原样输入到 Stata 中; 若使用斜体显示的部分, 则需要用户根据具体问题作相应地替换。


**例 A.1** 以数据集 *auto.dta* 为例介绍统计描述命令 *summarize*, 其一般的命令格式为:


```
1 . summarize [varlist] [if] [in] [weight] [, options]
```

```
1 . sysuse auto, clear           //读取数据
2 . summarize                   //输出所有变量的描述统计
3 . summarize mpg price, separator(1) //输出mpg和price的描述统计
4 //选项separator(1)表示每个变量的结果用直线分割
```

**注** Stata 对大小写敏感。此外, 在输入 Stata 命令时, 通常会使用缩写形式。例如, summarize 常缩写为 sum。

## do 文件

do 文件是 Stata 的命令程序文件, 其扩展名是 “.do”, 因此通常被称为 do 文件。对于简单的命令, 用户可以直接在命令窗口输入相关命令并执行, 但对于更加复杂的命令以及需要重复使用的命令, 更有效的方法是把所有的命令写在 do 文件里。用户可以通过单击 Stata 工具栏中的 do 文件按钮  来打开 do 文件编辑器, 也可通过菜单栏中的“窗口--do 文件编辑器--新 do 文件编辑器”打开, 打开后桌面会自动弹出 do 文件编辑器窗口。

在执行 do 文件时, 可以通过在命令窗口输入 `do filename` (结果窗口中显示命令执行过程) 或 `run filename` (结果窗口不显示命令执行过程) 一次性执行整个 do 文件的全部命令; 如果需要执行其中部分命令时, 则需要选中要执行的命令, 然后单击 do 文件编辑器工具栏的命令执行快捷键 , 或在 do 文件编辑器菜单栏中选择“工具--执行 (do)” (限 Windows 系统)。

在 do 文件中, 除了常用的命令外有时还需对命令添加注释, 常用的添加注释方法包括:

- 在某一行开头添加 \*, 此后整行内容都将被当作注释;
- 在某一行中使用 //, 这一行此后的内容将被当作注释;
- 使用配对的 /\* 和 \*/, 中间的内容将被当作注释;
- 在某一行中使用 ///, 该行 /// 之后的内容将被当作注释, 而该行下一行的内容将被视作注释前命令的一部分, 如果命令太长可使用该符号进行换行。

## 日志文件

在默认情况下, Stata 数据分析的结果是输出在结果窗口的。然而, 为了输出结果的可再现和冗长输出结果阅读的便利, 将输出结果保存在独立的文本文件中是一种可取的做法。此类独立的输出结果文件被称为日志 (log) 文件, 它保存着 Stata 运行的命令以及命令的输出结果。Stata 日志文件的默认扩展名是 “.log”, 但是由于 L<sup>A</sup>T<sub>E</sub>X 等软件也会生成以 log 为扩展名的文件, 建议创建日志文件时以 “.txt” 做后缀。此外, 日志文件通常与相应的 do 文件取成相同的文件名。

在 Stata 中, 创建日志文件可以采用如下命令格式:

```
1 . log using filename.txt, text replace
```

其中, *filename* 是用户设置的文件名; `replace` 作用是允许已经存在的同名日志文件被覆盖, 因此请慎用 `replace` 以免含有重要结果的日志文件被覆盖。当数据分析结束之后, 使用 `log close` 关闭日志文件。假设工作路径里已存在名为 `example` 的日志文件 `example.txt`, 其中保存了上次数据分析的命令和结果, 并且此日志文件处于关闭状态。如果想在此日志文件中追加输出结果, 可以使用如下命令:

```
1 . log using example.txt, append
```

此外，Stata 的工具栏提供了 log 按钮 ，可以实现对日志文件的快捷操作。

## Stata 命令结果的调用

只对数据进行分析但是不估计模型参数的 Stata 命令被称为 r-class 命令，此类命令的结果保存在 r() 中，可以通过 return list 查看，并通过 r(\*) 调用，\* 为具体的结果名称。例如：

```
1 . sysuse auto, replace
2 . sum mpg
3 . return list
4 . dis r(mean) //dis为display的缩写
```

估计参数的命令被称为 e-class 命令，例如，回归命令 regress 就是 e-class 命令。此类命令的结果保存在 e() 中，可以通过 ereturn list 查看，并通过 e(\*) 调用，\* 为具体的结果名称。例如：

```
1 . sysuse auto, replace
2 . reg mpg price weight //reg为regress的缩写，mpg为因变量，其余为自变量
3 . ereturn list
4 . dis e(r2) //显示回归的拟合优度
```

## A.2 Stata 编程基础

### 宏

在 Stata 编程中，宏 (macro) 是最基本也是最常用的工具之一，当一段内容需要被反复地使用时，定义宏能够使这段内容更方便地被调用。Stata 中的宏是一种临时变量，主要包括全局 (global) 宏和局部 (local) 宏，其中全局宏一旦被定义能够保留在 Stata 环境中直到退出 Stata 才消失，而局部宏只在定义它的程序之内有效，一旦程序运行完毕则会自动消失。

在 Stata 中，如果要定义全局宏，需要使用 global 命令，如果要定义局部宏，需要使用 local 命令。宏可以指代的内容包括字符、表达式的值和拓展函数的返回值，Stata 提供了很多宏的拓展函数，可通过 help extended\_fcn 来查看。

定义好宏之后就可以在后续程序中调用它们，使用局部宏的方法是用符号 ` 将宏的名称括起来，注意该符号左半部分的 “`” 位于 Tab 键之上，esc 键之下的 ~ 键，而右半部分是常见的单引号。使用全局宏的方法是在宏的名称之前添加符号 \$。可以使用 display 命令来显示宏所指代的内容。

```

1 ***** 定义宏并显示宏指代的内容 *****
2 . local a I love Stata      //定义局部宏a, 指代字符串 I love Stata
3 . display "`a'"            //输出 I love Stata
4 . global b I love Stata     //定义全局宏b, 指代字符串 I love Stata
5 . display "$b"             //输出 I love Stata
6 . local c = 2+2             //定义局部宏c, 指代表达式 2+2 的计算结果
7 . local d 2+2               //定义局部宏d, 指代字符串2+2
8 . local e: word count I love Stata
9 /*定义局部宏e, 指代拓展函数word count string的返回值*/
10 . display "`e'"            //输出3
11 . local f: word 1 of I love Stata
12 /*定义局部宏f, 指代拓展函数word # of string的返回值*/

```

**注** 当关闭上述 do 文件时, 在命令窗口输入 `display "`a'"` 结果为空; 但输入 `display "$b"` 仍然输出结果 `I love Stata`, 由此可见局部宏与全局宏的区别。

在实证研究中常常涉及到多个变量的回归分析, 此时使用宏来指代特定变量以重复使用能够提高编程效率。

**例 A.2** 以 Stata 自带的数据文件 `nlsw88.dta` 为例说明宏在回归中的使用, 该数据集包含了 1988 年采集的 2246 个美国妇女的相关资料, 提供了研究妇女工资的决定因素的相关数据, 主要包括小时工资 `wage`、每周工作时数 `hours`、种族 `race`、职业 `occupation`、年龄 `age`、是否大学毕业 `collgrad`、当前职业的工作年数 `tenure`、是否结婚 `married`、是否居住在南部地区 `south`、合计工作年数 `ttl_exp` 等变量。

```

1 ***** 用宏指代重复的命令片段 *****
2 . sysuse nlsw88, replace    //使用Stata系统自带的数据集nlsw88
3 . reg wage hours age tenure
4 . gen lnwage = log(wage)    //生成新的变量lnwage
5 . reg lnwage hours age tenure //线性回归, 因变量是lnwage, 自变量不变
6
7 /*由于两个回归方程中自变量没有发生变化, 可使用宏来简化命令*/
8 . local x hours age tenure
9 /*定义局部宏, 宏的名称为x, 指代变量为hours age tenure*/
10 . reg wage `x'
11 . reg lnwage `x'

```

在使用 `reg` 命令时, 让 Stata 汇报回归系数、使用稳健标准误、省略方差分析表的数据的对应选项分别是 `beta`、`vce(robust)`、`noheader`, 可以定义宏来指代这三个选项并在回归中使用:

```

1 . local option beta vce(robust) noheader
2 . reg wage `x', `option'
3 . reg lnwage `x', `option'

```

除此之外，宏的好处还在于可以批量修改命令：

```

1 . local x hours age ttl_exp //将自变量tenure修改为ttl_exp
2 . reg wage `x'
3 . reg lnwage `x'

```

## 条件函数

在编写程序时，经常会用到条件判断语句，Stata 提供了多种条件函数来解决条件判断问题，主要包括 `cond()`、`inrange()`、`inlist()`、`clip()`、`missing()` 等：

- `cond(ifc, a, b)`：当条件 ifc 为真时，返回  $a$ ，否则返回  $b$ ；
- `inrange(z, a, b)`：当  $a \leq z \leq b$  时，返回 1，否则返回 0；
- `inlist(z, a, b, ...)`：当  $z \in \{a, b, \dots\}$  时，返回 1，否则返回 0；
- `clip(x, a, b)`：当  $x \leq a$  时取  $a$ ，当  $a < x < b$  时取  $x$ ，当  $x \geq b$  时取  $b$ ；当  $x$  为缺失值时取缺失值；
- `missing()`：判断其自变量是否存在缺失值，存在取 1，否则取 0。

**例 A.3** 利用 Stata 自带数据集 `auto.dta` 说明上述条件函数的使用，此数据集含有 1978 年 74 辆汽车的数据，其中，变量 `rep78` 表示 1978 年维修次数，取值范围为 1 次到 5 次，并包含有少量缺失值，缺失值用 `.` 表示。这里需要注意 Stata 将 `.` 视为无穷大。

```

1 ***** 条件函数 *****
2 . sysuse auto, clear //读取数据集auto
3 . gen bad1 = cond(rep78>=4,1,0) //生成新变量bad1，当rep78>=4时取值为1
4 . replace bad1=. if rep78==. //rep78缺失时，bad1也缺失
5 . list rep78 bad1 in 1/20 //列出rep78和bad1的前20个元素
6
7 * 第3-4行命令效果等价于：
8 . gen bad2 = (rep78>=4) if rep78 !=.
9 * 或者，等价于：
10 . gen bad2 = cond(missing(rep78),.,cond(rep78>=4,1,0))
11
12 . gen bad3 = inrange(rep78, 1, 3) //生成bad3，当rep78属于[1,3]时取值为1
13 . list rep78 bad3 in 1/20
14
15 //生成bad4，当rep78的取值为1、2和4三者之一时bad4取值为1

```

```

16 . gen bad4 = inlist(rep78, 1, 2, 4)
17 . list rep78 bad4 in 1/20
18
19 //生成bad5, 将rep78的非缺失值截取在[1.5, 3.5]范围内
20 . gen bad5 = clip(rep78, 1.5, 3.5)
21 . list rep78 bad5 in 1/20
22
23 * 通常可以使用 drop if x==. 来删除x中的缺失值
24 * 使用missing可以批量删除多个变量的缺失值
25 . drop if missing(rep78, mpg, weight)

```

## 循环语句

Stata 中的循环语句主要包括 `forvalues`、`while` 和 `foreach` 三种，本部分将简单介绍以上三种循环语句的使用方法。

- `forvalues` 的命令格式为：

```

1 . forvalues lname = range{
2     包含`lname'的Stata命令
3 }

```

其中，`lname` 是局部宏，其取值由等号后面的 `range` 决定；`range` 用来设定数字序列的取值范围和变动方式。`forvalues` 在 `lname` 满足 `range` 设定的条件时，执行 {} 内包含 `lname` 的 Stata 命令。

- `while` 的命令格式为：

```

1 . while expr {
2     Stata命令
3 }

```

其中，`expr` 为逻辑表达式，当 `expr` 的判断结果为真时，`while` 执行 {} 中的命令。

- `foreach` 的第一种命令格式为：

```

1 . foreach lname in list{
2     包含`lname'的Stata命令
3 }

```

其中，`lname` 是局部宏，其取值需列在 `list` 中。此命令格式不需要指定 `list` 的取值类型。

- `foreach` 的第二种命令格式为：

```

1 . foreach lname of listtype list{

```



```

2     包含`lname'的Stata命令
3 }

```

如果 *lname* 后用 *of*, 则需要进一步指定 *listtype*, 可选项包括: *local* (局部宏)、*global* (全局宏)、*varlist* (变量名)、*newvarlist* (新的变量名) 和 *numlist* (数字)。

**例 A.4** 我们仍然使用 *auto.dta* 为例说明如何使用循环语句为变量添加标签, 以及对变量进行标准化处理, 关于标签的介绍请见下一节内容。本例用到的代码如下:

```

1  ***** 循环语句 *****
2  * 为新变量贴标签
3  . sysuse auto, clear
4  . drop if rep78==.
5  . tab(rep78), gen(worse)
6  . label define rep_status 0 "否" 1 "是" //定义值标签
7  . forvalues i = 1/5{
8  .     label var worse`i' "第`i'次维修" //`i'表示调用局部宏
9  .     label val worse`i' rep_status //对变量添加值标签
10 . }
11 . describe
12
13 * 使用while
14 . local i=1
15 . while `i'<= 5{
16 .     label var worse`i' "第`i'次维修"
17 .     label val worse`i' rep_status
18 .     local ++i
19 . }
20
21 * 对变量 weight length price 进行标准化处理
22 . foreach var in weight length price{
23 .     qui sum `var' //qui用来隐藏输出结果
24 .     gen std_`var' = (`var' - r(mean)) / r(sd) //`var'是局部宏
25 . }

```

**注** 上述代码中的第 5 行的 *tab* 是制表命令 *tabulate* 的缩写, 用来统计 *rep78* 的不同取值, 并汇报各个取值出现的次数、占比以及累积和, 选项 *gen* 的作用是根据 *rep78* 的不同取值生成相应的虚拟变量, 在机器学习中被称为 *one-hot* 编码。



## A.3 数据处理与 Stata 实现

在实证研究过程中，研究者从数据库或其他途径获取数据后，可以将数据调入 Stata 中进行处理，数据处理通常包括数据读入、数据预处理、数据分组与展示等过程。

### 数据读入

将数据读入 Stata 是进行数据处理和分析的第一步，Stata 可以直接读入的数据包括软件默认的“.dta”格式的数据、文本数据和 Excel 格式的数据，而无法直接读入的数据可以通过 Stat/Transfer 等数据转换软件转换成“.dta”格式的数据再读入 Stata。本附录将主要介绍如何应用 Stata 直接读入数据，关于 Stat/Transfer 软件的使用感兴趣的读者可自行学习。

Stata 读入数据的常用命令格式有：

- 读入 Excel 数据集，文件扩展名为 .xlsx

```
1 . import excel [using] filename [, options]
2 . import excel extvarlist using filename [, options]
```

第 1 行的命令格式用于读取整个文件，第 2 行的命令格式用于读取由 *extvarlist* 中变量构成的数据集子集。常用的选项有：

- `sheet("sheetname")`：指定读入的工作簿；
- `cellrange([start][:end])`：指定读入数据的范围；
- `firstrow`：将第一行视为变量名；
- `clear`：替换内存中的数据。
- 读入 Excel 数据集，文件扩展名为 .csv

```
1 . insheet [varlist] using filename [, options]
```

常用的选项有：

- `tab`：用 tab 分割的数据；
- `comma`：用逗号分割的数据；
- `[no]names`：第一行是否为变量名，是用 `names`，否则用 `nonames`；
- `clear`：替换内存中的数据。
- 读入扩展名为 .txt 的数据集

```
1 . infile varlist using filename [, options]
```

- 读入 Stata 格式 .dta 的数据集

```
1 . use filename [, clear nolabel]
2 . use [varlist] [if] [in] using filename [, clear nolabel]
```

第 1 行的命令格式用于读取整个文件，第 2 行的命令格式用于读取数据集的某个子集。选项 nolabel 限制读入数据的值标签。

除了使用上述命令之外，也可以通过 Stata 菜单栏中的文件--导入读取不同存储方式的数据集。


**例 A.5** 我们以数据文件 stockinfo.\* 为例说明数据读入命令的使用，\* 代表不同的文件扩展名，该数据集收集了 2011-2019 年中国 A 股市场非 ST 股票的主要公司财务数据，包括账面市值比 bm、杠杆率 lev、资产回报率 roa。本例用到的代码如下所示：

```

1 ***** 数据读入 *****
2 . cd "~/Stata Data Code" //设置当前工作路径
3 * 读入扩展名为.xlsx的数据集
4 . import excel "stockinfo.xlsx", sheet("Sheet1") firstrow clear
5 * 读入扩展名为.csv的数据集
6 . insheet using "stockinfo.csv", clear
7 * 读入扩展名为.txt的数据集
8 . infile str6 id str4 year bm lev roa using "stockinfo.txt", clear
9 * 读入扩展名为.dta的数据集
10 . use "stockinfo.dta", clear

```

**注** 第 8 行中的 str6 和 str4 的作用是将其后的变量定义为字符串，str 后面的数字表示字符串可容纳的最大长度。

除了读入已有的数据文件，用户还可以手动录入数据。在命令窗口输入 edit 或单击 Stata 工具栏中的数据编辑器按钮  都可以打开数据编辑器进行数据的录入。此外，用户可以使用 input 命令录入数据，例如：

```

1 ***** 数据录入 *****
2 . input str20 name age female income
3 .   "张三" 25 0 2500
4 .   "李四" 32 1 4500
5 . end

```

## 数据预处理

在完成数据导入后，接下来通常需要对数据进行预处理，包括对数据集和变量添加标签、删除重复观测值、处理缺失值、数据筛选和排序以及变量处理等操作。在完成以上步骤后，还可能涉及对数据结构和变量进行转换、数据分组和将多个数据集进行合并等。本小节将对如何利用 Stata 进行以上数据预处理操作进行逐一介绍。

## 变量命名和添加标签

当数据集中的变量名无法体现变量的具体含义时，通常需要将变量进行重命名，其具体命令格式为：

```
1 . rename varname newvarname
```

为了便于数据处理，数据集和变量命名通常较为简单，可以通过对数据集、变量和数值添加标签来进一步具体解释数据集、变量和数值的含义。添加标签的常用命令格式有：

- 对数据集添加标签：

```
1 . label data "newlabel"
```

- 对变量添加标签：

```
1 . label variable varname "newlabel"
```

- 对离散数据添加值标签 (value label)：

```
1 . label define labelname v1 "l1" v2 "l2"
2 . label values varname labelname
```

**例 A.6** 继续考虑例 A.5 的数据集，我们通过如下 Stata 命令为此数据集及其内部变量添加标签：

```
1 ***** 添加标签 *****
2 . use "stockinfo.dta", clear
3 * 为数据集添加标签
4 . label data "2011-2019年A股非ST公司主要财务指标"
5 * 为变量添加标签
6 . label var id "股票代码"
7 . label var year "年份"
8 . label var bm "账面市值比"
9 . label var lev "杠杆率"
10 . label var roa "总资产收益率"
11 * 为起始和结束年份添加值标签
12 . label define begin_end 2011 "第一年" 2019 "最后一年"
13 . label values year begin_end
14 . describe
```

数据集中各个变量的值标签名称和内容可以使用命令 `label list` 查看。

## 处理重复观测值和缺失值

有时因为重复录入等原因，数据集中可能存在重复的观测，这时应当先将其删除。Stata 处理重复观测的命令为 `duplicates`，下面举例说明此命令如何使用：

```

1 ***** 删除重复观测 *****
2 . use "stockinfo2.dta", clear
3 //报告重复观测情况，输出结果见图A.2左图
4 . duplicates report
5 //报告指定变量的重复观测情况，输出结果见图A.2右图
6 . duplicates report id year
7 . duplicates list           //显示重复的观测，输出结果见图A.3
8 . duplicates drop           //删除重复的观测
9 //或者使用duplicates drop id year, force

```

从第 3 行程序的输出结果来看，数据集中共有 15181 个唯一的观测，但是有 4 个重复观测，即两个观测各重复一次。

Duplicates in terms of all variables

Copies	Observations	Surplus
1	15181	0
2	4	2

Duplicates in terms of id year

Copies	Observations	Surplus
1	15181	0
2	4	2

图 A.2: Stata 报告重复观测命令的输出结果

Duplicates in terms of all variables

Group	Obs	id	year	bm	lev	roa
1	10	6	2011	.8794	.6529	.052165
1	11	6	2011	.8794	.6529	.052165
2	153	30	2018	1.0404	.3898	.074787
2	154	30	2018	1.0404	.3898	.074787

图 A.3: Stata 显示重复观测命令的输出结果

此外，数据集中往往不可避免地会存在缺失值，可以对数据集进行删除缺失值处理，或使用 `ipolate` 命令对缺失值进行线性插值填补。线性插值命令 `ipolate` 的格式为：

```
1 . ipolate yvar xvar [if] [in] , generate(newvar)
```

ipolate 利用 *yvar* 和 *xvar* 之间的线性函数对 *yvar* 的缺失值进行填补，并生成新的变量 *newvar* 保存填补后的数据。

下面举例说明：

```
1 ***** 处理缺失值 *****
2 . ss install missings, replace //安装missings程序包
3 . use "stockinfo3.dta", clear
4 . missings report //报告缺失值情况，见图A.4左图
5 . missings list //列出缺失值，见图A.4右图
6 . missings dropobs bm, force //删除缺失值
7 . missings dropobs lev, force
8 //或使用 drop if bm==.和 drop if lev==.
9 . ipolate bm year, gen(bm1) //利用线性插值填补bm的缺失值
10 . list id year bm bm1 in 89/91
```

从图 A.4 左图的输出结果看，bm 有两个缺失值，lev 有一个缺失值；图 A.4 右图的输出结果给出了缺失值的具体位置。第 9 行命令对 bm 的缺失值进行了线性插值填补并将填补之后的变量命名为 bm1。图 A.5 列出了填补之后数据集的第 89-91 行。

```
Checking missings in all variables:    Checking missings in all variables:
3 observations with missing values      3 observations with missing values
```

	#		bm	lev
bm	2	90.	.	.4636
lev	1	448.	.	.6885
		454.	.4134	.

图 A.4: Stata 报告和列出缺失值命令的输出结果

	id	year	bm	bm1
89.	20	2018	.2368	.79038569
90.	20	2019	.	.7308214
91.	21	2011	.8087	.66919551

图 A.5: Stata 线性插值填补缺失值命令的输出结果

## 数据筛选和排序

在数据预处理过程中，通常需要筛选出符合特定条件的数据并剔除不符合条件的数据。同时也可能会将数据按照一定顺序排列，以便观察数据的特征与规律。

首先举例说明数据的筛选：

```

1 ***** 数据筛选 *****
2 . use "stockinfo.dta", clear
3 . keep if id>=600000 & id<900000 //选择沪市主板A股公司
4 . keep if id>=1 & id<2000 //选择深市主板A股公司
5 . drop if id>=900000 //删除上证B股公司
6 . drop if id>=200000 & id<=300000 //删除深圳B股公司
7 . drop if id>=2000 & id<3000 //删除中小板公司
8 . drop if id>300000 & id<=400000 //删除创业板公司

```

再来看数据的排序, Stata 的常用排序命令为 `sort`, 其功能是对数据进行升序排列, 具体命令格式为:

```

1 . sort varlist [in] [, stable]

```

若在 `varlist` 中的变量个数是两个以上, 则先按照第一个变量升序排列, 在第一个变量的相同取值内部再将第二个变量升序排列, 以此类推; 当不使用 `stable` 选项时, 对于变量取值相同的观测, 排序是随机的; 使用选项 `stable` 的作用是按照原数据中的顺序排列变量取值相同的观测。

除了 `sort` 命令外, Stata 还提供了 `gsort` 命令实现更为复杂的数据排序, 其命令格式为:

```

1 . gsort [+|-] varname [[+|-] varname ...] [, generate(newvar) mfirst]

```

其中, `+` 表示升序排列, `-` 表示降序排列; `generate(newvar)` 根据变量排序之后的不同取值生成由 1, 2, 3, ... 构成的新变量用来表示不同的分组; `mfirst` 表示降序排列时将缺失值排在最前面。

下面的例子利用 `auto.dta` 数据集说明了排序命令的使用:

```

1 ***** 数据排序 *****
2 . sysuse auto, clear
3 . sort mpg weight, stable
4 . list mpg weight in 1/20
5 . gsort mpg, gen(group)
6 . gsort +mpg -price

```

## 系统变量

在数据处理时往往会用到 Stata 自带的一些系统变量 (system variable), 常用的系统变量包括:

- `_b[varname]`: 拟合模型中变量 `varname` 的系数;

- `_cons`: 直接使用表示 1; 间接使用表示截距项 (intercept);
- `_n`: 当前观测的序号;
- `_N`: 数据集中的总样本量, 或者当前 `by()` 分组中的样本数量;
- `_se[varname]`: 拟合模型中变量 `varname` 的标准误;
- `_pi`: 圆周率。

我们举例说明系统变量在数据处理中的用途:

```

1 ***** 系统变量的使用 *****
2 . sysuse auto, clear
3 . generate price2 = price[_n-1] //生成price的滞后一期变量
4 . regress price mpg foreign rep78
5 . display _b[mpg]                //显示mpg的回归系数-292.43424
6 . webuse dollhill12, clear        //加载Stata网站数据集
7 . by age (smokes), sort: generate wgt=pyears[_N]
8 //等价于bysort age (smokes): generate wgt=pyears[_N]

```

**注** 最后一行程序的功能是: 首先判断数据是否依 `age` 和 `smokes` 的顺序进行 `sort`, 然后按照 `age` 的不同取值进行分组, 生成新的变量 `wgt`, 并用每个分组的最后一个观测的 `pyears` 数据为其赋值。

## 变量类型转换

在数据处理过程中, 常常需要对变量的类型进行转换, 如将股票代码由数值型转换为字符型。数值型和字符型变量相互转换的命令有 `destring`、`tostring` 和 `real`, 下面举例说明两个命令的使用:

```

1 ***** 数值型和字符型变量的相互转换 *****
2 //将字符型变量转换成数值型变量
3 . clear
4 . input str6 stkcd str9 moutai
5 .      "000010" "600519.SH"
6 . end
7 . destring stkcd, generate(stkcd_num) //将stkcd转为数值型
8 //也可以使用real命令
9 . gen id = real(substr(moutai,1,6))   //substr用来提取字符串的子集
10 . list                                //id和stkcd_num都为数值型
11
12 // 将数值型变量转换为字符型变量
13 . clear
14 . input stkcd                          //此处的stkcd为数值型

```



```

15 . 10
16 . 600519
17 . end
18 . tostring stkcd, generate(stkcd_str)
19 . list                                     //stkcd_str为字符型

```

**注** 在第7行程序中，stkcd 转换为数值型之后用生成的新变量 stkcd\_num 保存。若直接用原变量名保存结果，可以使用 `destring stkcd, replace`；第9行用到的命令 `substr` 将在 A.3 的“字符型变量处理”小节中进一步进行介绍。

在分析时间序列数据时，经常涉及到数值型或字符型变量与日期型变量之间的转换。将字符型变量转换为日期型变量的常用 Stata 命令有：`date()`、`yearly()`、`halfyearly()`、`quarterly()`、`monthly()` 和 `weekly()`；将数值型变量转换为日期型变量的常用 Stata 命令有：`mdy()`、`yw()`、`ym()`、`yq()`、`yh()` 和 `year()`。为了说明上述命令的使用，我们引用 <http://www.princeton.edu/~otorres/> 提供的 Stata 教学课件 Time Series 上面的例子，具体代码如下：

```

1 ***** 字符型和数值型变量转换为日期型变量 *****
2 . use "http://www.princeton.edu/~otorres/Stata/date.dta", clear
3 . list in 1/3                                     //输出结果见图A.6左图
4 . gen datevar = date(date1,"DMY", 2022)           //DMY: 日月年
5 . format datevar %td                               // %td: 日度数据
6 . list in 1                                     //输出结果见图A.6右图
7 . gen datevar2 = date(date2,"MDY", 2022)
8 . format datevar2 %td
9 //date3情况比较复杂，Stata无法直接理解199511，可按如下方式处理：
10 . tostring date3, gen(date3a)
11 . gen len = length(date3a)
12 . gen year = substr(date3a,1,4)
13 . gen month = substr(date3a,5,1) if len == 6 //长度为6时第5位是月份
14 . gen day = substr(date3a,6,1) if len == 6    //长度为6时第6位是日
15 . replace month = substr(date3a,5,2) if len == 8
16 . replace day = substr(date3a,7,2) if len == 8
17 . destring month day year, replace
18 . gen datevar3 = mdy(month,day,year)           //利用mdy生成日期变量
19 . format datevar3 %td
20 . replace datevar3 = datevar3[_n-1] + 1 if datevar3 == .
21 . format datevar3 %tdCY-N-D                     //输出格式为1995-01-01
22
23 // 季度数据转换

```

```

24 . use tsdata.dta, clear
25 //数据来源: http://dss.princeton.edu/training/tsdata.dta
26 . gen date1=substr(date,1,7)
27 . gen datevar=quarterly(date1,"YQ")
28 . format datevar %tq

```

**注** 第 4 行的 2022 表示：当输入数据的年份由两位数字表示时，输出的年份的最大值不能超过 2022；在第 20 行，长度为 7 的字符串无法直接分清其代表的日期，这里的做法是将长度为 6 和 8 的情况处理好之后然后进行缺失值填补。

从上面的例子可以看出，`mdy()` 和 `yw()` 等命令是将保存在不同变量中的数值型年、月和日的数据整合为日期型变量数据；而 `date()` 和 `quarterly()` 等命令是根据字符型变量体现的时间特征，将其转换为相应的日期型变量数据。例如，含有形如 "1995h1" 的字符型变量可以用 `halfyearly()` 处理为半年数据，输出格式的设置命令为 `format varname %th`。

此外，用户可以分别用命令 `month()`、`day()` 和 `year()` 提取日期型变量的月、日和年信息。

	date1	date2	date3		date1	date2	date3	datevar
1.	1-Jan-95	1/1/1995	199511		1-Jan-95	1/1/1995	199511	01jan1995
2.	2-Jan-95	1/2/1995	199512					
3.	3-Jan-95	1/3/1995	199513					

图 A.6: 日期型变量转换示例中命令 `list` 的输出结果

## 长宽格式数据的转换

在存储面板数据时，我们既可以采用长 (long) 格式也可以采用宽 (wide) 格式。以宽格式存储的面板数据的每一个个体只有一条记录，不同时刻的观测位于不同的列；以长格式存储的面板数据的每一个个体都有多条记录，对应着不同的时刻。如图 A.7 所示，其中  $i$  表示个体， $j$  表示时间。

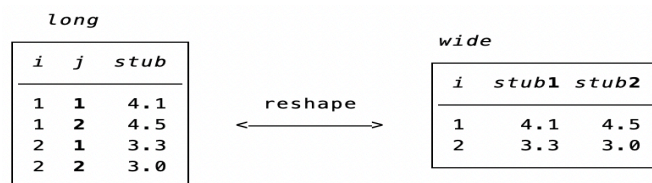


图 A.7: 长格式和宽格式数据之间的转换

在 Stata 中，宽格式数据和长格式数据之间的转换可以使用 `reshape` 命令，其命令格式为：

- 长格式转宽格式：`reshape wide stub, i(i) j(j)`；
- 宽格式转长格式：`reshape long stub, i(i) j(j)`，其中， $j$  为新变量。

以面板数据 `stockinfo.dta` 为例：

```

1 ***** 长宽格式数据转换 *****
2 . use "stockinfo.dta", clear
3 . reshape wide bm lev roa, i(id) j(year) //长格式转宽格式
4 . reshape long bm lev roa, i(id) j(year) //宽格式转长格式

```

## 新变量创建

在数据处理过程中，常会基于已有变量创建新的变量，这就涉及到 Stata 提供的数据运算命令，常用命令如表 A.1 所示。在 Stata 中，`generate` 可用来生成数据运算产生的新变量，此命令可以简写为 `gen` 或 `g`。

代数运算		逻辑运算		比较关系		特殊函数	
+	加	!	非	>	大于	exp	指数
-	减	~	非	<	小于	log	对数
*	乘		或	>=	不小于	^	乘方
/	除	&	与	<=	不大于	sqrt	平方根
				==	相等	l.	滞后一期
				!=	不等于	f.	提前一期
				~=	不等于	d.	一阶差分
						s.	季节差分

表 A.1: Stata 常用运算符

命令 `egen` 是 `gen` 的拓展，用来生成更复杂的变量。新产生的变量会被自动加入到数据集的最后一列，可以通过 `order` 命令来改变数据集中变量的排列顺序，其命令格式为：

```

1 . order varlist [, options]

```

其中，常用的选项有：

- `first`：将 `varlist` 中所列的变量移至数据集开头；
- `last`：将 `varlist` 中所列的变量移至数据集结尾；
- `before(varname)`：将 `varlist` 中所列的变量移至 `varname` 之前；
- `after(varname)`：将 `varlist` 中所列的变量移至 `varname` 之后，

更多关于 `order` 用法的介绍可以使用命令 `help order` 打开其帮助文档查看。这里给出一个简单的例子：

```

1 ***** 新变量创建和排序 *****
2 . use "stockinfo.dta", clear
3 . egen maxbm = max(bm), by(id) //生成变量maxbm，取值为相同id中bm的最大值
4 . order maxbm roa, after(bm) //将maxbm和roa移到bm后

```

## 随机数生成

在统计模拟和 Bootstrap 时，我们通常需要利用计算机生成服从特定分布的随机数，Stata 中提供了多种概率分布的随机数生成函数，具体可参见 [help function](#) 中的 [random-number functions](#)。下面以均匀分布和正态分布等常用分布为例说明如何利用 Stata 生成随机数，具体代码如下：

```

1 ***** 生成随机数 *****
2 . clear
3 . set obs 1000           //设置样本量
4 . set seed 202208        //设置随机种子
5 . gen u1 = runiform()    //生成(0,1)区间上的均匀分布随机数
6 . gen u2 = runiform(-1,1) //生成(-1,1)区间上的均匀分布随机数
7 . gen n1 = rnormal()     //生成服从标准正态分布的随机数
8 . gen n2 = rnormal(0,2)  //生成期望为0标准差为2的正态分布随机数
9 . gen c1 = rchi2(2)      //生成自由度为2的卡方分布随机数
10 . gen t1 = rt(2)        //生成自由度为2的t分布随机数

```

## 字符型变量处理

对于字符型变量的处理通常包括连接、提取和匹配。Stata 中的字符连接与代数运算类似，同一字符串的连接可以用 `*`，两个字符串之间的连接可以用 `+`。字符串子集的提取可以使用命令 `substr`，其命令格式为：

```
1 . substr(s,n1,n2)
```

其中，`s` 为 ASCII 字符串，`n1` 是提取字符串子集的起始位置，`n2` 是要提取的长度。字符串的匹配可以使用命令 `regexm`，其命令格式为：

```
1 . regexm(s,re)
```

其中，`s` 为字符串，`re` 为正则表达式 (regular expression)，若 `s` 包含 `re` 或满足 `re` 要求，则 `regexm` 返回 1，否则返回 0。

下面看一个简单的例子：

```

1 ***** 字符型变量的处理 *****
2 . use "stockinfo.dta", clear
3 . tostring id, gen(idstr)
4 . replace idstr="0"*(6-length(idstr)) + idstr
5 // 字符连接，对不足6位的idstr前面用0补齐至6位
6

```

```

7 . tostring year, gen(yearstr)
8 . gen year2 = substr(yearstr,3,2)    //提取年份数据的后两位
9
10 . gen searchid = 1 if regexm(idstr, "002")
11 . replace searchid = 0 if searchid == .
12 // 生成新变量searchid, 股票代码里含有002时取1, 否则取0
13
14 . gen searchSH = 1 if regexm(idstr, "^6")
15 . replace searchSH = 0 if searchSH == .
16 // 生成新变量searchSH, 股票代码以6开始时取1, 否则取0

```

除了 `regexm` 命令之外, `regex` 类命令中还有 `regexr` 和 `regexs`, 感兴趣的用户可以使用命令 `help regex` 进一步了解。值得注意的是 `regex` 类命令只适用于 ASCII 字符编码, 而对于 Unicode 编码的字符串, 则需要使用 `ustrregex` 类命令, 较为常用的函数有匹配函数 `ustrregexm`、替换函数 `ustrregera` 和截取函数 `ustrregexs`。此外, Stata 还提供了大量的处理字符串数据的字符串函数, 感兴趣的用户可以通过命令 `help string functions` 进一步了解。

我们再通过一个例子说明常用的字符串函数的使用:

```

1 ***** 字符型变量的处理 *****
2 . clear
3 . input str30 x
4 .   "Programming&STATA 202208"
5 .   "Financial Econometrics"
6 . end
7 . list
8
9 * 匹配: 匹配变量中的"STATA", 匹配成功返回1, 否则返回0
10 . gen j1 = ustrregexm(x, "STATA")
11
12 * 匹配变量中的"STATA"并提取, 0表示提取满足条件的整个字符串
13 . gen j2 = ustrregexs(0) if ustrregexm(x, "STATA")
14
15 * 替换: 寻找"STATA"中的任何一个元素, 如果有就替换为"R"
16 . gen j3 = ustrregera(x, "[STATA]", "R")
17
18 * 替换: 将非"STATA"中的字符替换为空, 最后只留下"STATA"
19 . gen j4 = ustrregera(x, "[^STATA]", "")
20

```

```

21 * 字符串函数
22 . gen t1 = strupper(x)          //将所有字母转换成大写
23 . gen t2 = strlower(x)         //将所有字母转换成小写
24 . gen t3 = strproper(x)        //将指定位置的字母大写
25 . gen len = strlen(x)          //计算字符长度
26 . gen count = wordcount(x)     //统计由空格隔开的词数量
27
28 * 分割
29 . split x, parse(" ") gen(test)
30 //将x中的字符串按照空格分割，并生成命名方式为test#的新变量，#代表1, 2, ...
31 . list
32
33 * 字符搜索
34 . capture drop t*              //删除以t开头的变量
35
36 * 提取字符串中的第一个单词
37 . gen t1 = ustrregexs(0) if ustrregexm(x, "\w+")

```

**注** 第 24 行的命令 `strproper` 的作用是使字符串中第一次出现的字母大写，紧随非字母字符之后的字母大写，其余字母小写；第 37 行命令中出现的 `"\w+"` 为匹配普通字符（如字母、数字）的正则表达式，关于正则表达式的介绍可以参考 <https://www.runoob.com/regexp/regexp-syntax.html>。常用的正则表达式有：

- `[abc]`: 匹配 a 或 b 或 c；
- `[^abc]`: 匹配 a、b 和 c 之外的任意字符；
- `[a|b]`: 匹配 a 或 b；
- `[a-z]`: 匹配所有小写字母；
- `[a-zA-Z]`: 匹配所有大小写字母；
- `[0-9]`: 匹配任一数字；
- `\w`: 匹配任一字母、数字、下划线和汉字；
- `\s`: 匹配任何空白字符，包括空格、制表符、换页符等；
- `+`: 匹配前面的子表达式一次或多次，要匹配 `+`，使用 `\+`；
- `*`: 匹配前面的子表达式零次或多次，要匹配 `*`，使用 `\*`；
- `^`: 匹配输入字符串开始的位置；
- `$`: 匹配输入字符串结尾的位置。

## 因子变量

在实证研究中，常常需要在回归模型中加入**虚拟变量**、**交乘项**、**平方项**或**高次项**，在 Stata 中可以使用**因子变量**（**factor variable**）用于简化构造这些变量的操作。因子变量是

对现有变量的延伸，能够从类别变量中生成虚拟变量、构造类别变量之间的交乘项、类别变量与连续型变量之间的交乘项或连续变量之间的交乘项（或多项式），常用的因子变量运算符包括以下五种：

- `i.`：设置类别变量不同水平的示性变量（indicator）的一元运算符；
- `c.`：将变量视为连续变量的一元运算符；
- `o.`：省略一个连续变量或类别变量的某一水平示性变量的一元运算符；
- `#`：设置交乘项的二元运算符；
- `##`：设置主项和交乘项的二元运算符。

**注** 对于因子变量运算符，有如下几点需要说明：

1. 由因子变量运算符生成的示性变量和交乘项在数据集中不会被显示出来；
2. 类别变量的值必须是非负整数，范围在  $[0, 32740]$  之间；
3. 因子变量运算符可以与 `l.` 和 `f.` 等特殊函数一起使用。

想进一步了解因子变量的用户可使用命令 `help fvvarlist` 调出 Stata 的帮助文档。

再次考虑例 A.2，在研究妇女工资的决定因素问题时研究者可能需要进一步构造虚拟变量或交乘项。在 `nlsw88.dta` 数据集中，小时工资 `wage`、每周工作时数 `hours`、年龄 `age`、当前职业的工作年数 `tenure`、合计工作年数 `ttl_exp` 为连续型变量；种族 `race`、职业 `occupation` 为类别变量；是否大学毕业 `collgrad`、是否结婚 `married`、是否居住在南部地区 `south` 为虚拟变量。

在研究是否为黑人对妇女工资的影响时，需要生成是否为黑人的虚拟变量，具体命令如下所示：

```
1 ***** 因子变量的使用 *****
2 . sysuse nlsw88, clear
3 . des race           //查看race值标签名称，为racelbl
4 . label list racelbl  //显示race值标签的赋值方式，black为2
5 . gen black = 2.race  //生成black虚拟变量，黑人为1，否则为0
6 . gen lwage = log(wage)
7 . reg lwage black
```

若我们希望直接得到虚拟变量 `black` 的估计系数，而无需储存这个变量，则可以直接使用因子变量运算符生成虚拟变量并参与回归：

```
1 . reg lwage 2.race
```

若想要研究种族对妇女工资的影响，则需要根据 `race` 的不同取值生成相应的虚拟变量，注意 `race` 的不同取值有 3 个，但是虚拟变量只需要两个，否则会有完全共线性问题。此问题的回归命令为：

```
1 . reg lwage i.race
```



生成虚拟变量的过程中，如需改变基准组可使用 `ib`，或简写为 `b`，常见用法有：

- `ib*.`：使用 `*` 作为基准组，`*` 为. 后面的变量的某一水平的取值；
- `ib(#*)`：使用有序类别变量的第 `*` 个水平作为基准组；
- `ib(first)`：使用类别变量的最小值所对应的水平作为基准组；
- `ib(last)`：使用类别变量的最大值所对应的水平作为基准组；
- `ib(freq)`：使用类别变量出现频率最高的值所对应的水平作为基准组；
- `ibn.`：无基准组。

除虚拟变量外，回归中还往往需要加入变量的交乘项或高次项，Stata 提供了 `#` 和 `##` 实现这些变量的构造，例如：

```
1 * 加入种族race和职业类别occupation的交乘项
2 . reg lwage i.race#i.occupation
3 * 除了交乘项之外也放入主项
4 . reg lwage i.race##i.occupation
5 * 加入种族race和当前职业的工作年数tenure的交乘项
6 . reg lwage i.race#c.tenure
7 * 加入年龄age及其平方项
8 . reg lwage c.age##c.age
```

在面板固定效应模型中，可以利用因子变量加入年度、公司或行业等固定效应。

**例 A.7** 以数据集 `stockltd.dta` 为例，该数据集收集了 2011-2019 年中国 A 股市场非 ST 股票的主要数据，包括账面市值比 `bm`、杠杆率 `lev`、资产回报率 `roa`、表示上市公司所属不同行业的类别变量 `indu`、表示是否为国企的虚拟变量（国企为 1），以及股票的左尾崩盘风险 `ltd`。我们的目的是研究 `bm` 和 `lev` 对 `ltd` 的影响，并分别控制个体效应、行业效应和年度效应，具体代码如下：

```
1 ***** 面板数据的固定效应 *****
2 . use "stockltd.dta", clear
3 * 加入公司、年度固定效应
4 . reg ltd bm lev i.id i.year
5 //或使用xtreg命令：
6 . xtset id year //设置面板数据的个体和时间变量
7 . xtreg ltd bm lev i.year, fe
8 * 加入行业、年度固定效应
9 . reg ltd bm lev i.indu i.year
```

## 数据分组

在实证研究中,常常需要对数据进行分组处理,在 Stata 中通常使用函数 `by` 或 `bysort` 实现分组功能,它们的命令格式为:

```
1 . by varlist1 [(varlist2)] [, sort rc0]: stata_cmd
2 . bysort varlist1 [(varlist2)] [, rc0]: stata_cmd
```

在 `by` 命令中的 `sort` 选项的作用是使 `by` 命令按照 `varlist` 中变量的顺序对数据进行 `sort`; `rc0` 的作用是即使 `stata_cmd` 在某一分组里报错也让该命令在剩余的分组执行; `bysort` 命令实际上是使用了 `sort` 选项的 `by` 命令。

继续考虑例 A.7 中的数据集合,我们说明分组命令的用法:

```
1 ***** 数据分组 *****
2 . use "stockltd.dta", clear
3 . bysort id (year): gen growth=(bm-bm[_n-1])/bm[_n-1]
4
5 // 分年度按roa升序将公司分组,使每组的公司数为20
6 . bysort year: egen roarank = rank(roa)
7 . gen group20 = int((roarank-1)/20) + 1
8
9 // 分组进行计算
10 * 计算不同代码股票ltd的累积和
11 . bysort id: gen rsum=sum(ltd)
12 * 计算不同代码股票ltd的累积乘积
13 . bysort id: gen rprod = exp(sum(ln(ltd)))
```

**注** 第 3 行程序的作用是判断数据集是否依 `id` 和 `year` 的顺序 `sort`,若是,按照 `id` 的不同取值将数据集进行分组并在组内计算增长率 `growth`。

除了利用 `by` 和 `bysort` 命令进行分组操作之外,还可以利用 `cut` 命令、循环语句等方式实现分组操作,例如:

```
1 ***** 数据分组 *****
2 . use stockltd.dta, clear
3 * 将2011年的所有公司按roa升序分为20组
4 . egen rankroa = cut(roa) if year==2011, group(20)
5 . tab rankroa //统计每组公司个数、占比和累积和
6
7 * 分年度将所有公司按roa升序分为20组
8 . gen rankroa1 = .
9 . forvalues i=2011/2019{
```

```

10 . egen temp = cut(roa) if year==`i' & roa!=., group(20)
11 . replace rankroa1 = temp if year==`i' & roa!=.
12 . drop temp
13 . }
14 . replace rankroa1 = (rankroa1+1) //使第一组组号为1而不是0
15
16 * 生成虚拟变量mroa, 当roa大于或等于其中位数时, mroa为1, 否则为0
17 . sscc install todummy, replace
18 . todummy roa, median gen(mroa)

```

## 数据合并

在数据处理中, 通常需要对不同数据集中的数据进行匹配与合并。在 Stata 中, 数据合并通常用到的命令包括 `merge` (横向合并)、`append` (纵向合并); 匹配常用到的命令包括 `reclink` (字符串模糊匹配)、`joinby` (多对多匹配) 等。

利用 `merge` 进行合并时要求两组数据中必须至少有一个共同变量, 同时, `merge` 可以进行 `1:1` (一对一匹配)、`1:m` (一对多匹配)、`m:1` (多对一匹配) 和 `m:m` (多对多匹配)。在匹配后, 生成的 `_merge = 1` 表示正在使用的数据, `_merge = 2` 表示合并的数据, `_merge = 3` 表示成功合并的数据, 因此需要保留 `_merge = 3`, 并将新生成的变量 `_merge` 删掉。请看下面的例子:

```

1 ***** 数据横向合并 *****
2 . use "stockltd.dta", clear
3 . preserve //保存原始数据
4 . keep id year bm lev
5 . save d1.dta, replace //第一份数据, 包括id、year、bm和lev
6 . restore //恢复原始数据
7 . keep id year roa ltd
8 . save d2.dta, replace //第二份数据, 包括连续变量roa和ltd
9
10 . use "d1.dta", clear
11 * 从数据集d2.dta中合并roa数据
12 . merge 1:1 id year using d2.dta, keepusing(roa)
13 . keep if _merge == 3 //保留匹配成功的数据
14 . drop _merge //将创建的新变量_merge删除
15
16 * 从数据集indu.dta中合并行业类别, 多对一匹配
17 . merge m:1 id using indu.dta
18 . keep if _merge == 3

```

```

19 . drop _merge
20 . list in 1/3, clean noobs    //无分割线，无观测值序号

```

横向合并用于为数据集添加新的变量。若要在数据集中添加新的观测数据，则可以使用 `append` 进行纵向合并。在纵向合并过程中，`append` 要求两组数据具有全部相同的变量，否则就会出现缺失值。下面举例说明 `append` 的用法：

```

1  ***** 数据纵向合并 *****
2  . use "stockltd.dta", clear
3  . preserve
4  . keep if _n <=5
5  . keep id year bm lev
6  . save d3.dta, replace
7  . restore
8
9  . keep if _n >=5 & _n <=10
10 . keep id year bm lev
11 . save d4.dta, replace
12
13 . use d3.dta, clear
14 . append using d4.dta        //纵向合并
15 . list, clean noobs

```

在进行横向合并时，如果匹配变量在内容上存在微小差别，可以使用非官方命令 `reclink` 进行模糊匹配，`reclink` 命令可解决的匹配问题包括大小写不同、部分字母缺漏或冗余、以及顺序颠倒等，其命令格式为：

```

1 . reclink varlist using filename, idmaster(varname) idusing(varname)
   gen(newvarname)

```

其中，`idmaster(varname)` 用来设定主数据集 (master dataset) 中可以唯一识别不同观测的变量名；`idusing(varname)` 用来设定合并数据集 (using dataset) 中可以唯一识别与 `idmaster` 相似的不同观测的变量名；注意 `idmaster` 和 `idusing` 使用的 `varname` 必须不同。此外，`gen(newvarname)` 也是必选项，用来设定保存匹配得分的新变量名称。匹配得分越高，匹配效果越好。

下面举例说明 `reclink` 的使用：

```

1  ***** 数据模糊匹配 *****
2  . ssc install reclink, replace
3  . clear

```

```

4 . input str13 name str14 city
5 .   "Han meimei" "Shanghai"
6 .   "Li Lei"     "Chengdu"
7 .   "Zhang, san" "Beijing"
8 .   "Si, Li"     "Shenzhen"
9 . end
10 . gen id1=_n
11 . save name1.dta, replace
12
13 . clear
14 . input str14 name str10 city
15 .   "Han, meim"  "shaanghai"
16 .   "LI lei"     "chengdou"
17 .   "zhangSan"   "beijiang"
18 .   "Li si"      "shezhen"
19 . end
20 . gen id2=_n
21 . save name2.dta, replace
22
23 * 模糊匹配
24 . use name1.dta, clear
25 . reclink name city using name2.dta, idmaster(id1) idusing(id2)
   . gen(matchscore)
26 . list, clean noobs

```

在进行多对多匹配时，相较于 merge，joinby 的匹配效果更好，因此被广泛使用。请看下面例子：

```

1 ***** 命令joinby的使用 *****
2 . clear
3 . input group str3 x1
4 .   1  "A"
5 .   1  "B"
6 .   1  "C"
7 .   1  "D"
8 . end
9 . save d5.dta, replace
10
11 . clear

```

```

12 . input group str3 x2
13 .   1  "M"
14 .   1  "M"
15 . end
16 . save d6.dta, replace
17
18 . use d5.dta, clear
19 . joinby group using d6.dta
20 . list, clean noobs

```

## A.4 Stata 绘图

在数据分析和实证研究中常常需要进行数据和回归结果的可视化，在 Stata 中，可通过 `graph` 命令进行绘图，关于该命令的用法可以使用命令 `help graph` 调出 Stata 帮助文档进一步了解更多详情。

### 双坐标轴 X-Y 图

常见的双坐标轴 X-Y 图的绘制可以通过 `twoway` 命令实现，其中 `scatter`（散点图）和 `line`（线图）子命令较为常用。本小节使用 <https://data.princeton.edu/wws509/data/sets/> 提供的 `effort.dta` 数据集对绘图命令做简要地介绍，该数据由 Mauldin 和 Berelson 收集，主要包括对拉丁美洲 20 个国家的社会环境指数（social setting）、家庭生育项目努力指数（family planning program effort index）和 1965 年至 1975 年期间毛出生率下降百分比（the percent decline in the crude birth rate (CBR)）的观测，具体代码如下：

```

1 ***** 双坐标绘图 *****
2 . use effort.dta, clear
3 * 散点图：Y：生育率变化 (change)；X：社会环境 (setting)
4 . graph twoway scatter change setting
5
6 * 散点图和回归拟合线：使用命令 lfit 可以进行回归并绘制样本回归线
7 . graph twoway (scatter setting effort) (lfit setting effort)
8
9 * 样本回归线添加置信带：使用命令 lfitci
10 * ytitle(): 为y轴指定标签；legend(off): 隐藏图例
11 . graph twoway (lfitci setting effort) (scatter setting effort), ///
12   ytitle("Fertility Decline") legend(off)
13
14 * 使用 mlabel 添加点标签

```

```

15 . graph twoway (lfitci change setting) ///
16     (scatter change setting, mlabel(country))
17 * 处理点标签重叠情况
18 . gen pos = 3 //默认3点钟方向
19 . replace pos = 11 if country == "TrinidadTobago" //移动到11点
20 . replace pos = 9 if country == "CostaRica" //移动到9点
21 . replace pos = 2 //移动到2点
22     if country == "Panama" | country == "Nicaragua"
23 . graph twoway (lfitci change setting) ///
24     (scatter change setting, mlabel(country) mlabv(pos) )
25
26 * 进一步调整：加title、修改坐标轴标签、修改图例位置等
27 . graph twoway (lfitci change setting) ///
28     (scatter change setting, mlabel(country) mlabv(pos) ), ///
29     title("Fertility Decline by Social Setting") ///
30     ytitle("Fertility Decline") ///
31     legend(ring(0) pos(5) order(2 "linear fit" 1 "95% CI"))
32 . graph export fig1.pdf, replace

```

**注** 第 31 行的 legend 是用来设置图例参数的选项，这里用到了如下设置：

- ring(0)：表示将图例移动到绘图区域内；
- pos(5)：表示将图例框放置在 5 点钟位置附近；
- order(2 "linear fit" 1 "95% CI")：表示按照该顺序标记第二个和第一个做图项目。

上述命令最后绘制的结果如图 A.8 所示：

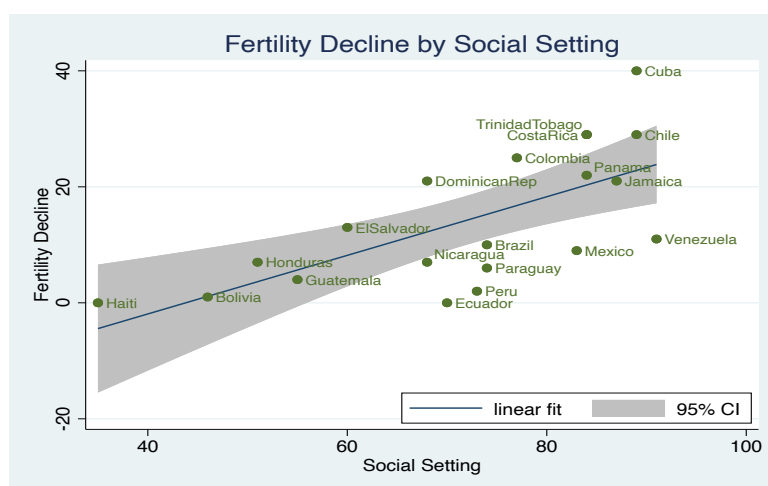


图 A.8: 拉丁美洲 20 个国家生育率下降百分比与社会环境的散点图和样本回归线

在 Stata 绘图中，通常使用 `title()` 和 `subtitle()` 来添加标题，分别通过 `legend()`、`note()`、`caption()` 添加图例、注释和图形附带说明，更多关于图表标题和图例的 Stata 绘图命令可进



一步参考由 [help title\\_options](#) 调出的帮助文档, 关于图形中的文本命令可参考由 [help graph text](#) 调出的帮助文档。对于坐标轴缩放比例和范围的控制常用的命令有 [xscale\(\)](#) 和 [yscale\(\)](#); 常用的控制刻度 (tick) 和标签的命令包括 [xtick\(\)](#)、[ytick\(\)](#)、[mtick\(\)](#)、[ylabel\(\)](#) 和 [xlabel\(\)](#), 更多相关命令可参考由 [help axis\\_scale\\_options](#) 和 [help axis\\_label\\_options](#) 调出的帮助文档。

接下来, 我们介绍几种常见的双坐标轴图的 Stata 实现:

## 线图

使用 Stata 自带的标普 500 股价数据集 `sp500.dta` 来说明线图的绘制:

```

1 ***** 线图绘制 *****
2 . sysuse sp500, clear
3 * 简单线图: 开盘价 (open)、收盘价 (close) 对观测序号做线图
4 . gen n =_n                                //n为当前观测的序号
5 . graph twoway line open close n in 1/50    //只画前50天的走势
6
7 * 添加标题和图例, 并将图例移动到绘图区域7点钟位置
8 . graph twoway line open close n in 1/50, sort clpattern(- 1)    ///
9     title("SP500 Stock Price") subtitle("January 2 to March 14")    ///
10    xtitle("Date") ytitle("Price")    ///
11    legend( order(1 "open" 2 "close") ring(0) pos(7))
12
13 * 更改线条颜色和粗细
14 . graph twoway (line open close n in 1/50, sort clpattern(- 1)    ///
15     clcolor(blue red) clwidth(0.9 0.9)),    ///
16     title("SP500 Stock Price") subtitle("January 2 to March 14")    ///
17     xtitle("Date") ytitle("Price")    ///
18     legend( order(1 "open" 2 "close") ring(0) pos(7))
19
20 * 利用scheme改变图形风格, 使用命令help scheme查看帮助文档
21 . graph display, scheme(economist)    //使用经济学人杂志绘图风格
22 . graph export fig2.pdf, replace

```

**注** 第 7 行中的 `sort` 选项设定数据要按照横坐标排序; `clpattern()` 用来设置线的类型, 常用的参数有: `1` 表示实线, `dot` 表示点线 (dotted line), `-` 表示虚线等; 第 14 行中的选项 `clcolor` 用来改变线条颜色, `clwidth` 用来改变线条粗细。

最后绘制出的经济学人杂志风格的 S&P 500 股价走势如图 A.9 所示:

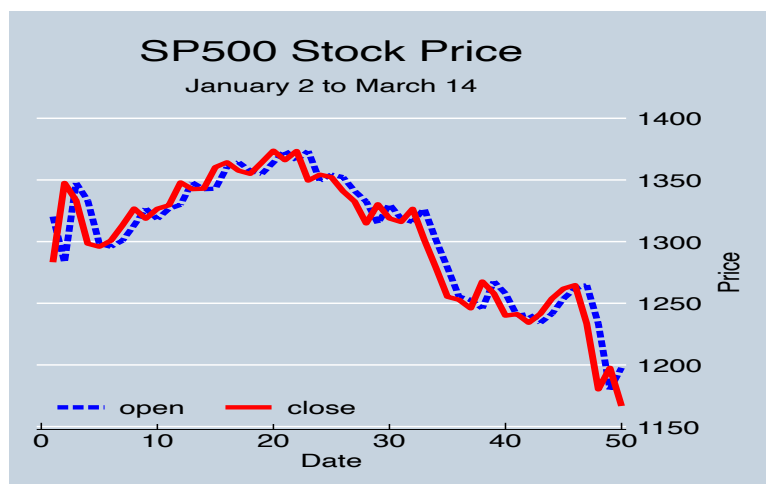


图 A.9: S&amp;P 500 股价走势

## 直方图

直方图能够展示变量的频数分布，我们利用 2019 年我国上市公司资产收益率数据来说明 Stata 如何绘制直方图，具体程序如下：

```

1 ***** 直方图绘制 *****
2 . use "stockltd.dta", clear
3 . drop if roa==. | year < 2019 //剔除缺失值并保留2019年上市公司roa数据
4 . ssc install winsor2, replace
5 . winsor2 roa, replace cut(1 99) //进行1%和99%的缩尾处理
6 . sum(roa) //查看描述性统计，确定横轴范围
7 * 绘制直方图
8 . histogram roa, fraction normal ///
9     title ("2019年上市公司资产收益率直方图") ///
10    start(-0.5) xlabel (-0.5 (0.1) 0.2) xmtick(##2) ///
11    xtitle("上市公司资产收益率") ytitle("所占比重")
12 . graph export bar1.pdf, replace

```

**注** 第 10 行中的 `xmtick(##2)` 设定横轴每隔两个单位显示一个刻度；第 8 行中的 `fraction` 设定绘制直方图时纵轴显示比例，`normal` 的作用是在直方图上添加正态分布的 pdf。

最终绘制出的 2019 年我国上市公司资产收益率直方图如图 A.10 所示：

## 箱线图

箱线图能够展示变量的分布特征，图 A.11 给出了箱线图每个部分含义的说明，此图片来源为：<https://www.kdnuggets.com/2019/11/understanding-boxplots.html>。

我们利用我国上市公司 2011-2019 年账面市值比数据来说明 Stata 如何绘制箱线图，具体程序如下：

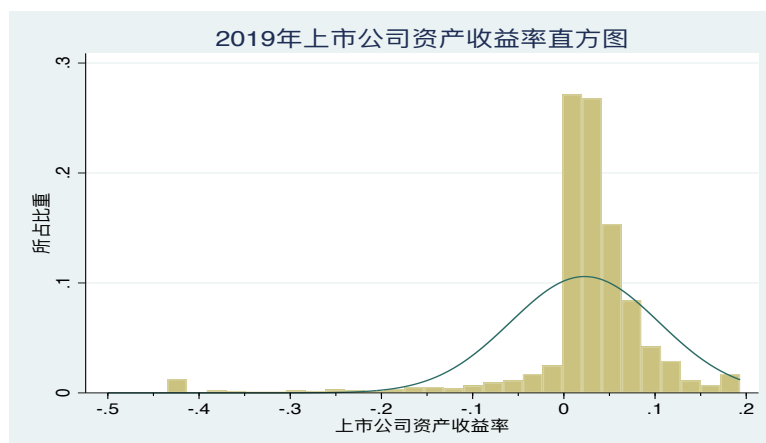


图 A.10: 2019 年上市公司资产收益率的直方图

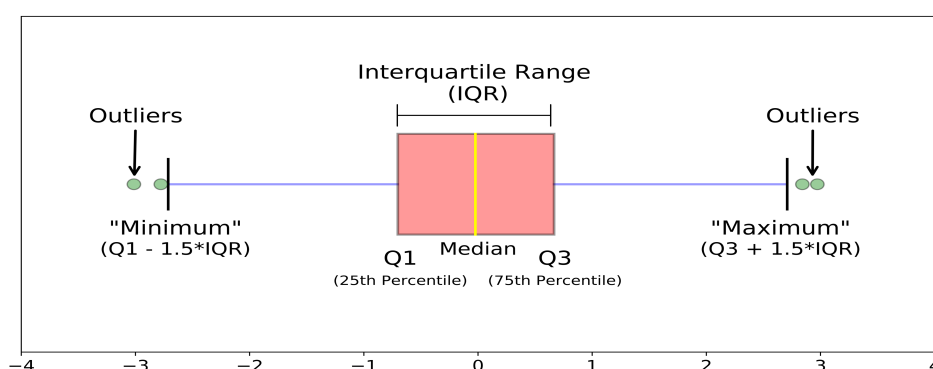


图 A.11: 箱线图组成部分图解

```

1 ***** 箱线图绘制 *****
2 . use "stockltd.dta", clear
3 . drop if bm ==.
4 . winsor2 bm, replace cut(1 99)
5 * 绘制箱线图
6 . graph box bm, over(year, label(angle(45))) ///
7     box(1, color("51 102 204")) ylabel(0(0.2)1.2) nooutside ///
8     title("上市公司账面市值比箱线图") ymtick(##2) ytitle("账面市值比")
9 . graph export box1.pdf, replace

```

**注** 第 7 行中的 `box(1, ...)` 表示 `graph box` 命令后第一个变量的箱线图；`color` 通过 RGB 值将箱线图的颜色设置为蓝色；`nooutside` 的作用是设置绘制箱线图时不显示异常值。

最终绘制出的我国上市公司 2011-2019 分年度账面市值比箱线图如图 A.12 所示：

## 核密度估计

核密度估计也是对变量分布可视化的一种方式，与直方图相比，核密度估计曲线更加的平滑（smooth）。在 Stata 中可以使用 `kdensity` 进行核密度估计并绘图。利用 Stata 自带的数据集 `citytemp.dta`，我们说明如何使用 `kdensity` 命令，具体代码如下：

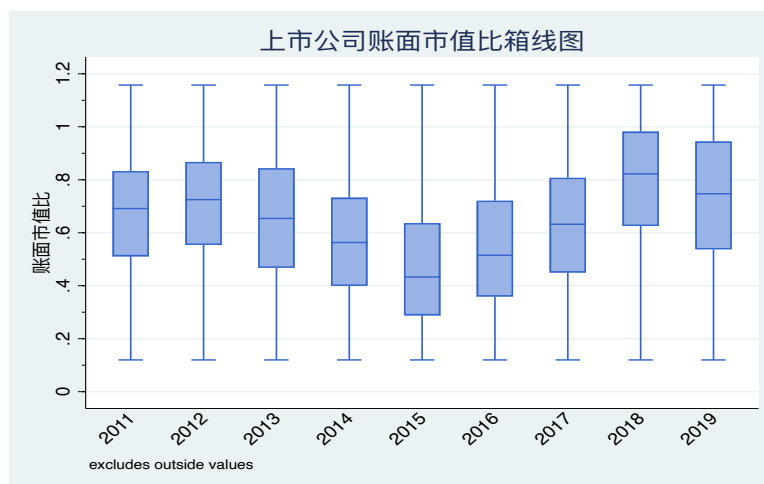


图 A.12: 上市公司账面市值比箱线图

```

1 ***** 核密度估计 *****
2 . sysuse citytemp, clear
3 * 对每个region的1月温度分别进行核密度估计并保存结果
4 . forvalues i=1/4 {
5 .     capture drop x`i' d`i'
6     //检查x`i'和d`i'是否已经被定义
7 .     kdensity tempjan if region== `i', generate(x`i' d`i')
8     //核密度估计, 用x`i'保存格点, d`i'保存格点处的核密度估计
9 . }
10 . gen zero = 0
11
12 * 绘制range plot with area shading
13 . graph twoway rarea d1 zero x1, color("blue%20") ///不透明度达到20%
14     || rarea d2 zero x2, color("green%20")      ///
15     || rarea d3 zero x3, color("orange%20")     ///
16     || rarea d4 zero x4, color("red%20")        ///
17     title(January Temperatures by Region)      ///
18     ytitle("Smoothed density")                 ///
19     legend(ring(0) pos(2) col(1) order(2 "NC" 1 "NE" 3 "S" 4 "W"))
20 . graph export kernel1.pdf, replace

```

**注** 绘制带阴影的区域图的命令格式为:

```

1 . twoway rarea y1var y2var xvar [if] [in] [, options]

```

其含义是以 *xvar* 为横轴, 绘制 *y1var* 和 *y2var* 之间区域的阴影图。第 19 行中的 *col(1)* 用来将不同区域的图例排成一列。

最终绘制出的不同地区 1 月份温度的核密度估计如图 A.13 所示：

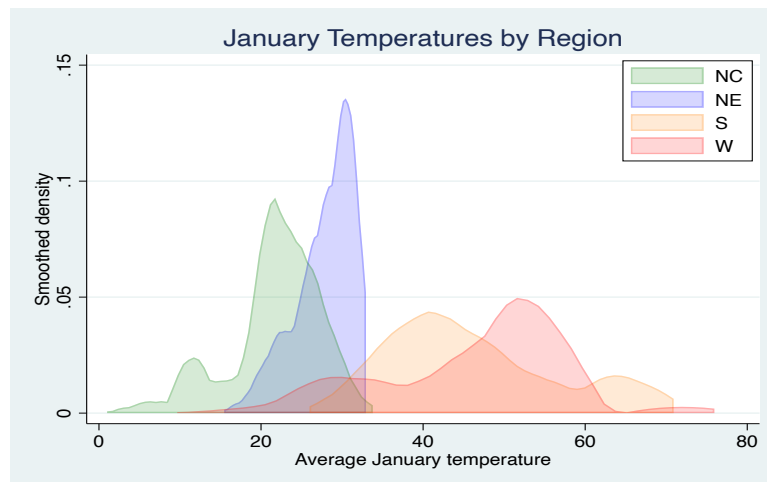


图 A.13: 不同地区 1 月份温度核密度估计

## A.5 Stata 结果输出

在使用 Stata 进行实证研究并撰写论文的时候，涉及到将 Stata 的分析结果输出到 Word 或 L<sup>A</sup>T<sub>E</sub>X，常用的结果输出命令有四种，包括 `esttab`、`xxx2docx`、`outreg2` 和 `asdoc`，其中，`xxx2docx` 包括用于输出描述性统计结果的 `sum2docx`、用于输出两样本均值比较的  $t$  检验结果的 `t2docx`、用于输出相关系数矩阵的 `corr2docx` 以及用于输出回归结果的 `reg2docx`。本节首先介绍如何使用上述四种命令输出描述性统计结果，之后主要介绍如何使用 `xxx2docx` 和 `esttab` 命令输出不同类型的分析结果，其中包括利用 `esttab` 命令输出 L<sup>A</sup>T<sub>E</sub>X 表格的相应方法。

### 输出描述统计结果

汇报描述统计是数据处理的一个重要环节，描述统计的作用主要包括：有助于理解数据类型及分布、识别异常值，识别缺失值等。Stata 提供了多种汇报描述统计的命令，主要包括 `asdoc sum`、`sum2docx`、`esttab` 配合 `estpost sum`，以及 `outreg2`，其中，`sum2docx` 和 `outreg2` 为非官方命令，使用前需要事先安装：

```
1 . ssc install sum2docx, replace
2 . ssc install outreg2, replace
```

我们以 `nlsw88.dta` 数据集为例说明 Stata 输出描述统计结果的方法：

```
1 ***** 描述统计结果输出 *****
2 . sysuse nlsw88, clear
3 . local varlist "wage age race married grade collgrad south union
   occupation"
4 * asdoc: 不支持中文，字符型变量会报错
```

```

5 . asdoc sum `varlist',                               ///
6     save(Myfile1.doc) replace                         ///
7     stat(N mean sd min p50 max) dec(3)               /// dec(3): 保留小数点后三位
8     title(asdoc_Table: Descriptive statistics)
9
10 * sum2docx: 支持中文, 字符型变量会报错, 能设置每个统计量的小数位数
11 . sum2docx `varlist' using Myfile2.docx, replace ///
12     stats(N mean(%9.2f) sd(%9.3f) min(%9.2f)        ///
13     median(%9.2f) max(%9.2f))                        ///
14     title(sum2docx_Table: Descriptive statistics)
15
16 * outreg2: 不支持中文, 字符型变量在输出结果中自动剔除
17 . outreg2 using Myfile3, sum(detail) replace word ///
18     eqkeep(N mean sd min p50 max) ///
19     fmt(f) keep(`varlist')                          ///fmt(f): 使用固定长度的字符串
20     sortvar(wage age grade)                          ///变量排序
21     title(outreg2_Table: Descriptive statistics)
22
23 * esttab: 不支持中文, 有字符型变量会直接运行并输出空白结果, 能设置小数位数
24 . estpost summarize `varlist', detail
25 . esttab using Myfile4.rtf, ///
26     cells("count mean(fmt(2)) sd(fmt(2)) min(fmt(2)) p50(fmt(2))
27     max(fmt(2))") ///
28     noobs compress replace title(esttab_Table: Descriptive statistics)
29 // 输出.tex文件
30 . esttab using Myfile4.tex, ///
31     cells("count mean(fmt(2)) sd(fmt(2)) min(fmt(2)) p50(fmt(2))
32     max(fmt(2))") ///
33     noobs compress replace ///
34     title(esttab_Table: Descriptive statistics) ///
35     booktabs page(array, makecell) alignment(cccccc) width(\hsize)

```

**注** 在使用 `sum2docx` 命令时, 用户可以设置要汇报的每个统计量的小数点位数, 例如, 在第 12 行中的 `%9.2f` 表示宽度为 9 个字符且小数点后有两位的一般格式; 第 27 行中的 `noobs` 表示不显示样本数量, `compress` 表示压缩表格横向宽度; 第 33 行中列出的 `esttab` 输出  $\text{\LaTeX}$  表格的选项含义为:

- `booktabs`: 在导言区添加 `\usepackage{booktabs}`, 输出表格 (三线表格);
- `page(packages)`: 创建完整的  $\text{\LaTeX}$  文档, 在导言区加入 `packages` 中列出的包;

- `alignment(cccccc)`: 定义从第二列开始的列对齐方式，默认居中；
- `width(\hsize)`: 使表格宽度延伸至页面宽度。
- `fragment`: 只输出创建  $\text{\LaTeX}$  表格的代码，不能与 `page` 同时使用。

### 输出两样本 $t$ 检验结果

检验两个样本的均值是否相等的  $t$  检验是常用的统计检验，Stata 中的命令 `estpost ttest` 配合 `esttab` 可以方便地输出两样本  $t$  检验的结果。此外，非官方命令 `t2docx` 也可以实现两样本  $t$  检验结果的呈现，使用前需要事先安装：

```
1 . ssc install t2docx, replace
```

我们继续以 `nlsw88.dta` 数据集为例说明 Stata 输出两样本  $t$  检验结果的方法：

```
1 ***** 两样本t检验结果输出 *****
2 . sysuse nlsw88, clear
3 . local varlist "wage age race married grade collgrad union"
4 * t2docx命令的使用:
5 . t2docx `varlist' using Myfilet1.docx, replace ///
6     not by(south) title(t2docx_Table: T_test by group)
7
8 * esttab命令的使用:
9 . estpost ttest `varlist', by(south)
10 . esttab using Myfilet2.rtf,                ///
11     cells("N_1 mu_1(fmt(3)) N_2 mu_2(fmt(3)) b(star fmt(3))") ///
12     starlevels(* 0.10 ** 0.05 *** 0.01)    ///
13     noobs compress replace title(esttab_Table: T_test by group)
14 // 输出.tex文件
15 . esttab using Myfilet2.tex,                ///
16     cells("N_1 mu_1(fmt(3)) N_2 mu_2(fmt(3)) b(star fmt(3))") ///
17     starlevels(* 0.10 ** 0.05 *** 0.01)    ///
18     noobs compress append title(esttab_Table: T_test by group)///
19     booktabs page width(\hsize)
```

**注** 第 6 行中的 `not` 表示不显示  $t$  统计量，`by(south)` 表示按照 `south` 分组；第 12 行中的 `starlevels` 可以设置依据  $p$  值标注星号的临界值；第 18 行中的 `append` 表示将结果追加到之前已经存在的文件中，如果不想追加保存可以使用 `replace`。



## 变量间相关系数矩阵的输出

若回归中的解释变量之间存在高度的线性相关性，回归系数估计量的统计性质往往将受到影响。因此，汇报变量之间的相关系数矩阵常被视为一种回归变量选择的方法。Stata 中的命令 `estpost correlate` 配合 `esttab` 可以方便地输出变量之间的相关系数矩阵。此外，非官方命令 `corr2docx` 也可以汇报变量之间的相关系数矩阵，使用前需要事先安装：

```
1 . ssc install corr2docx, replace
```

以 `nlsw88.dta` 数据集为例，我们说明如何利用 Stata 输出变量间的相关系数矩阵：

```
1 ***** 相关系数矩阵的输出 *****
2 . sysuse nlsw88, clear
3 . local varlist "wage age race married grade collgrad"
4 * corr2docx: 可汇报Pearson相关系数，不报告p值，但是可标注显著性的星号
5 . corr2docx `varlist' using Myfilecorr1.docx, replace ///
6     spearman(ignore) pearson(pw) star ///
7     title(corr2docx_Table: correlation coefficient matrix)
8
9 * esttab: 能报告p值，可以设置标注显著性星号的临界值
10 . estpost correlate `varlist', matrix
11 . esttab using Myfilecorr2.rtf, ///
12     unstack not noobs compress nogaps replace ///
13     star(* 0.1 ** 0.05 *** 0.01) ///
14     b(%8.3f) p(%8.3f) ///
15     title(esttab_Table: correlation coefficient matrix)
16 // 输出.tex文件
17 . esttab using Myfilecorr2.tex, ///
18     unstack not noobs compress nogaps append ///
19     star(* 0.1 ** 0.05 *** 0.01) ///
20     b(%8.3f) p(%8.3f) ///
21     title(esttab_Table: correlation coefficient matrix) ///
22     booktabs page width(\hsize)
```

**注** 第 6 行中的 `spearman(ignore)` 表示不汇报 Spearman 相关系数，`pearson(pw)` 表示展示所有变量两两之间的 Pearson 相关系数，`star` 表示按默认方式为显著性标注星号；第 12 行中的 `unstack` 表示结果分列汇报（不堆叠在一列），`not` 表示不显示  $t$  统计量，`nogaps` 表示不采用竖线分割各列。

## 回归结果的输出

实证分析往往涉及多个回归结果之间的比较，利用 Stata 制作清晰且易于读者比较的回归结果表格是实证研究者应该掌握的基本技能。请看下面使用 `nlsw88.dta` 数据集的例子：

```

1 ***** 回归结果的输出 *****
2 . sysuse nlsw88.dta, clear
3 . tabulate race, gen(race_num) //生成类别变量race对应的虚拟变量
4 . drop race_num1
5 . gen lwage = log(wage)
6 . qui reg lwage age married occupation
7 . est store m1 //保存回归结果到m1
8 . qui reg lwage age married collgrad occupation
9 . est store m2
10 . qui reg lwage age married collgrad occupation race_num*
11 . est store m3
12 . esttab m1 m2 m3 using Myfilereg.rtf, ///
13     replace star( * 0.10 ** 0.05 *** 0.01 ) ///
14     nogaps compress order(married) drop(occupation) ///
15     b(%20.3f) se(%7.2f) r2(%9.3f) ar2 aic bic ///
16     obslast scalars(F) indicate("race=race_num*") ///
17     mtitles("OLS-1" "OLS-2" "OLS-3") ///
18     title(esttab_Table: regression result)
19 // 输出.tex文件
20 . esttab m1 m2 m3 using Myfilereg.tex, ///
21     replace star( * 0.10 ** 0.05 *** 0.01 ) ///
22     nogaps compress order(married) drop(occupation) ///
23     b(%20.3f) se(%7.2f) r2(%9.3f) ar2 aic bic ///
24     obslast scalars(F) indicate("race=race_num*") ///
25     mtitles("OLS-1" "OLS-2" "OLS-3") ///
26     title(esttab_Table: regression result) booktabs page width(\hsize)

```

**注** 第 14 行的 `order(married)` 是将 `married` 作为第一个解释变量汇报结果，`drop(occupation)` 表示不汇报 `occupation` 的系数；第 15 行的 `b`、`se`、`r2`、`ar2`、`aic` 和 `bic` 分别表示汇报系数估计、标准误、 $R^2$ ，调整的  $R^2$ ，AIC 和 BIC；第 16 行的 `obslast` 表示把样本量汇报在最后一行，`scalars(F)` 表示汇报系数联合显著性检验的 F 统计量，`indicate("race=race_num*")` 表示在回归结果表格中显示控制了 `race` 类别变量，`race_num*` 是指 `race_num2` 和 `race_num3`，这里不能使用 `i.race` 进行回归，原因是 `indicate` 需要数据集中有 `race_num2` 和 `race_num3`

这些变量；第 17 行的 `mtitles` 设定了回归表格中三列回归结果的列名。

表 A.2: `esttab`\_Table: regression result

	(1) OLS-1	(2) OLS-2	(3) OLS-3
married	-0.022 (0.02)	-0.022 (0.02)	-0.042* (0.02)
age	-0.004 (0.00)	-0.003 (0.00)	-0.004 (0.00)
collgrad		0.506*** (0.03)	0.494*** (0.03)
_cons	2.182*** (0.15)	2.093*** (0.14)	2.169*** (0.14)
race	No	No	Yes
$R^2$	0.040	0.176	0.181
adj. $R^2$	0.039	0.174	0.179
AIC	3783.023	3445.551	3434.481
BIC	3805.875	3474.115	3474.472
F	31.376	118.817	82.187
N	2237	2237	2237

Standard errors in parentheses

\*  $p < 0.10$ , \*\*  $p < 0.05$ , \*\*\*  $p < 0.01$

使用上述命令生成的文件 `Myfilereg.tex`，我们得到回归表格 A.2。

## 致谢

此附录初稿的撰写由上海财经大学金融学院在读博士生赵天瑀完成，在此表示感谢。

## 参考文献

- 毛新述, 2022. 实证研究方法论——Stata 应用 [M]. 北京: 中国人民大学出版社.
- 许琪, 2021. Stata 数据管理教程 [M]. 北京: 北京大学出版社.